

FILE ID**PLIHEEP

PPPPPPPP	LL	IIIIII	HH	HH	EEEEEEEEE	EEEEEEEEE	PPPPPPPP	
PPPPPPPP	LL	IIIIII	HH	HH	EEEEEEEEE	EEEEEEEEE	PPPPPPPP	
PP	PP	LL	II	HH	EE	EE	PP	PP
PP	PP	LL	II	HH	EE	EE	PP	PP
PP	PP	LL	II	HH	EE	EE	PP	PP
PP	PP	LL	II	HH	EE	EE	PP	PP
PPPPPPPP	LL	II	HHHHHHHHHHHH	EEEEEEEEE	EEEEEEEEE	PPPPPPPP		
PPPPPPPP	LL	II	HHHHHHHHHHHH	EEEEEEEEE	EEEEEEEEE	PPPPPPPP		
PP	LL	II	HH	HH	EE	EE	PP	
PP	LL	II	HH	HH	EE	EE	PP	
PP	LL	II	HH	HH	EE	EE	PP	
PP	LL	II	HH	HH	EE	EE	PP	
PP	LLLLLLLLLL	IIIIII	HH	HH	EEEEEEEEE	EEEEEEEEE	PP	
PP	LLLLLLLLLL	IIIIII	HH	HH	EEEEEEEEE	EEEEEEEEE	PP	

LL			SSSSSSSS
LL			SSSSSSSS
LL		II	SS
LL		II	SSSSSS
LL		II	SSSSSS
LL		II	SS
LLLLLLLLLLL			SSSSSSSS
LLLLLLLLLLL			SSSSSSSS

(1) 64 pli\$aloheap - allocate heap space
(1) 118 pli\$freeheep - deallocate heap space
(1) 157 pli\$allocation - return number of time that has been allocated

```
0000 1 .title pli$heep - pl1 runtime heap allocation
0000 2 .ident /1-003/
0000 3
0000 4 ****
0000 5 *
0000 6 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 * ALL RIGHTS RESERVED.
0000 9
0000 10 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 * TRANSFERRED.
0000 16 *
0000 17 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 * CORPORATION.
0000 20 *
0000 21 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 *
0000 24 *
0000 25 ****
0000 26
0000 27 ++
0000 28 facility:
0000 29
0000 30 VAX/VMS PL1 runtime library.
0000 31
0000 32 abstract:
0000 33
0000 34 Runtime routines to allocate and deallocate heap space.
0000 35
0000 36 author: R. Heinen 16-dec-1978
0000 37
0000 38 modifications:
0000 39
0000 40 1-002
0000 41 Modified both routines, to allow them be called with different
0000 42 number of parameters and so work for the controlled variable.
0000 43 Alex Wu 03/24/82
0000 44
0000 45 add the new routine - PLI$ALLOCATION to return the number of
0000 46 generation that a controlled variable has been allocated.
0000 47 Alex Wu 03/30/82
0000 48
0000 49 1-003 Bill Matthews 29-September-1982
0000 50
0000 51 Invoke macros $defdat and rtshare instead of $defopr and share.
0000 52
0000 53
0000 54
0000 55 external definitions
0000 56
0000 57
```

PLI\$HEEP
1-003

- pl1 runtime heap allocation

H 6

16-SEP-1984 02:21:23 VAX/VMS Macro V04-00
6-SEP-1984 11:38:35 [PLIRTL.SRC]PLIHEEP.MAR;1

Page 2
(1)

0000 58 :
0000 59 : local data
0000 60 :
0000 61 :
0000 62 rtshare

```

0000 64 .sbttl pli$aloheap - allocate heap space
0000 65 :++
0000 66 pli$aloheap - allocate heap space
0000 67
0000 68 functional description:
0000 69
0000 70 This routine allocates a memory block using "lib$get_vm".
0000 71 The allocated block is a longword (or two longword)-bigger in order to
0000 72 save the allocated size (and/or the previous pointer) in the block itself.
0000 73 The size is stored in the first longword (the pointer is save in the first
0000 74 longword). The returned address
0000 75 is the address of the actual free space.
0000 76
0000 77 inputs:
0000 78
0000 79 (ap) = 2 (or 3 for the controlled variables)
0000 80 4(ap) = longword size
0000 81 8(ap) = address to store address of the data
0000 82 12(ap) = address to the current block -- if exist
0000 83
0000 84 outputs:
0000 85
0000 86 r0 = success indicator fixed binary(32)
0000 87
0000 88 r0 = ss$normal for success
0000 89 lib$insvirmem - insufficient memory for request
0000 90 lib$badblosiz - bad size parameter
0000 91
0000 92 If an error is indicated by r0 then the returned address is 0.
0000 93 and the error is signalled.
0000 94 --
0000 95 .entry pli$aloheap,0
0000 96
0000 97
0000 98
0000 99
0000 100
0000 101
0000 102 10$: clrl a8(ap) ; assume allocation failure
0000 103 addl3 #4,4(ap),-(sp) ; build following arg list
0000 104 cmpl #2,(ap) ; based ?
0000 105 beql 10$ ; less than or equal then yes
0000 106 addl2 #4,(sp) ; one more longword for pointer
0000 107 clrl -(sp) ; address to return address
0000 108 pushab (sp)
0000 109 pushab 8(sp)
0000 110 calls #2,g^lib$get_vm ; allocate the memory
0000 111 blbc r0,20$ ; if low clear then error
0000 112 popl r0 ; get address of allocated space
0000 113 popl (r0)+ ; insert size in buffer
0000 114 cmpl #2,(ap) ; based ?
0000 115 beql 15$ ; less than or equal than yes
0000 116 movl 12(ap),(r0)+ ; save pointer
0000 117 movl r0,a8(ap) ; return address of allocated memory
0000 118 ret
0000 119 pushl r0 ; signal error condition
0000 120 calls #1,g^lib$signal
0000 121 ret

```

```

0041 118 .sbttl pli$freeheap - deallocate heap space
0041 119 :++
0041 120 :+ pli$freeheap - deallocate heap space
0041 121 :
0041 122 : functional description:
0041 123 :
0041 124 : This routine is the complementary routine to "pli$alocheep".
0041 125 : The memory is returned via "lib$free_vm".
0041 126 :
0041 127 : inputs:
0041 128 :
0041 129 :     (ap) = 1 (or 2 for the controlled variables)
0041 130 :     4(ap) = address of memory
0041 131 :     8(ap) = address of the based pointer -- if exist
0041 132 :
0041 133 : The block must have been allocated using "pli$alocheep".
0041 134 :
0041 135 : outputs:
0041 136 :
0041 137 :     For the controlled variables, there is a side effect on the
0041 138 :     based pointer @8(ap)
0041 139 :     r0 = success indicator fixed binary(32)
0041 140 :
0041 141 :     r0 = ss$ normal for success
0041 142 :     lib$badblosiz - incorrect block size parameter
0041 143 :-- :
0000 0041 144 .entry pli$freeheap,0
0000 0043 145 :
50 04 AC 04 C3 0043 146 subl3 #4,4(ap),r0      : address block size or pointer longword
6C 01 D1 0048 147 cmpl #1,(ap)                 : based variable ?
08 BC 07 13 004B 148 beql 30$                   : less than or equal than yes
50 60 DD 004D 149 movl (r0),@8(ap)             : save previous block pointer
50 04 C2 0051 150 subl2 #4,r0                  : adress to block size
50 DD 0054 151 30$: pushl r0                    : put address in memory
6E 9F 0056 152 pushab (sp)                   : set up arg list
50 DD 0058 153 pushl r0                    :
00000000'GF 02 FB 005A 154 calls #2,g^lib$free_vm : free the memory
04 0061 155 ret

```

0062 157 .sbttl pli\$allocation - return number of time that has been allocated
0062 158 :++
0062 159 : pli\$allocation - return number of time that has been allocated
0062 160 :
0062 161 : functional description:
0062 162 :
0062 163 : This routine perform the builtin function ALLOCATION which return the
0062 164 : generations that a controlled variable has been allocated. It walks
0062 165 : through the link list and increment the count each time looks at a
0062 166 : new block until the end of the list (null pointer).
0062 167 :
0062 168 : inputs:
0062 169 :
0062 170 : (ap) = 2
0062 171 : 4(ap) = address of the current active block
0062 172 : 8(ap) = address of the return value
0062 173 :
0062 174 : The block must have been allocated using "pli\$aloheep".
0062 175 :
0062 176 : outputs:
0062 177 :
0062 178 : 8(ap) contains the return value
0062 179 :--
0000 0062 180 .entry pli\$allocation,0
0064 181
50 08 BC D4 0064 182 clrl a8(ap) : initialize the counter
04 AC D0 0067 183 movl 4(ap),r0 : get the pointer
08 13 006B 184 beql 30\$: : if null pointer then done
50 08 BC D6 006D 185 10\$: incl a8(ap) : increment the counter
F8 70 D0 0070 186 movl -(r0),r0 : get previous block address
12 0073 187 bnequ 10\$: : keep looping if not null pointer
04 0075 188 30\$: ret
0076 189
0076 190 .end

PLISHEEP Symbol table

- pl1 runtime heap allocation

L 6

16-SEP-1984 02:21:23 VAX/VMS Macro V04-00
6-SEP-1984 11:38:55 [PLIRTL.SRC]PLIHEEP.MAR;1

Page 6
(1)

LIB\$FREE VM	*****	X	01
LIB\$GET VM	*****	X	01
LIB\$SIGNAL	*****	X	01
PLISALLOCATION	00000062	RG	01
PLISALOCHEEP	00000000	RG	01
PLISFREEHEEP	00000041	RG	01

+-----+
! Psect synopsis !
+-----+

PSECT name

Allocation PSECT No. Attributes

PLISCODE 00000000 (0.) 00 (0.) NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
00000076 (118.) 01 (1.) PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Phase	Page faults	CPU Time	Elapsed Time
Initialization	9	00:00:00.05	00:00:00.59
Command processing	68	00:00:00.54	00:00:01.51
Pass 1	61	00:00:00.55	00:00:01.26
Symbol table sort	0	00:00:00.00	00:00:00.00
Pass 2	43	00:00:00.38	00:00:00.60
Symbol table output	1	00:00:00.01	00:00:00.01
Psect synopsis output	1	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	183	00:00:01.56	00:00:04.00

The working set limit was 750 pages.

2560 bytes (5 pages) of virtual memory were used to buffer the intermediate code.

There were 10 pages of symbol table space allocated to hold 6 non-local and 6 local symbols.

190 source lines were read in Pass 1, producing 17 object records in Pass 2.

1 page of virtual memory was used to define 1 macro.

+-----+
! Macro library statistics !
+-----+

Macro library name

Macros defined

~~-\$255\$DUA28:[PLIRTL.OBJ]PLIRTMAC.MLB;1
-\$255\$DUA28:[SYS1IB1STARLET.MLB;2~~

101

TOTALS (all libraries)

3 GETS were required to define 1 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=TRACEBACK/LIS=LIS\$:PLIHEEP/OBJ=OBJ\$:PLIHEEP MSRC\$:PLIHEEP/UPDATE=(ENH\$:PLIHEEP)+LIB\$:PLIRTMAC/LIB

0308 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PLIFORMAT
LIS

PLIGETBUF
LIS

PLIGETEDI
LIS

PLIHEEP
LIS

PLIMSGTXT
LIS

PLIPUTFIL
LIS

PLIRMSBIS
LIS

PLIRECPT
LIS

PLIREAD
LIS

PLIREWRT
LIS

PLIOPEN
LIS

PLIPROTEC
LIS

PLIPUTEDI
LIS

PLIGETLIS
LIS

PLIPKDIUL
LIS

PLIPUTLIS
LIS

PLIMSGPTR
LIS

PLIPKDIUS
LIS

PLIPUTBUF
LIS

PLIGETFIL
LIS